


METODOLOGIA ÁGIL

Lílian Simão Oliveira

- 
- Fonte:
 - Pressman, 2004
 - Aulas Prof. Auxiliadora Freire e Sabrina Schürhaus
 - Alexandre Amorin

Chaos Report 2013



Por quê????



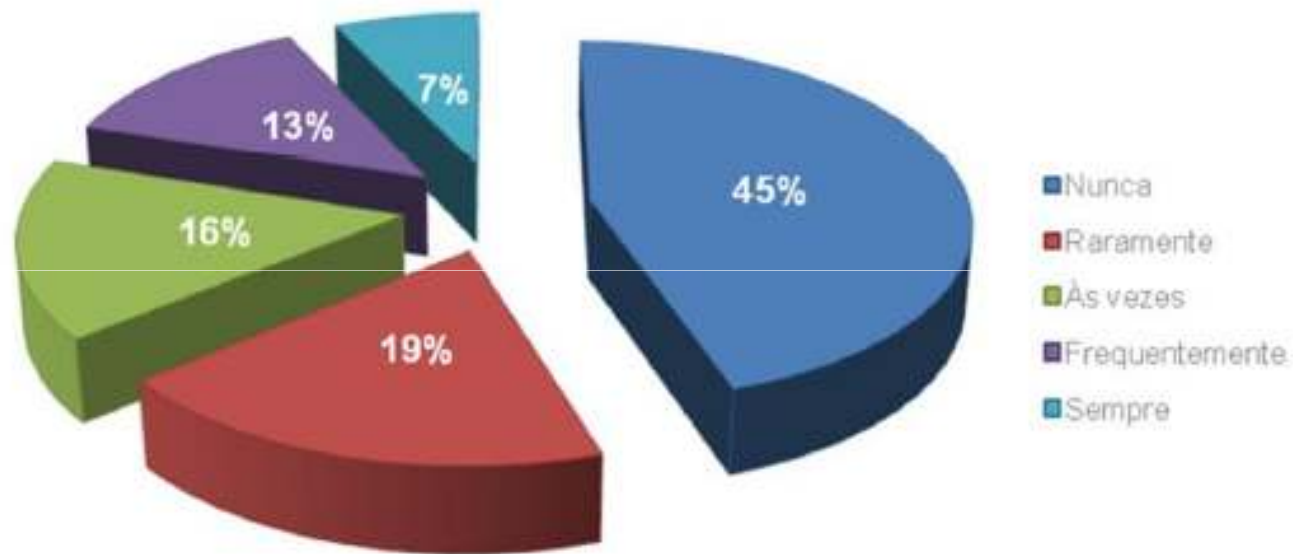
Principais Causas



- A comunicação entre as partes envolvidas nos projetos é muito fraca;
- A visibilidade do andamento real nos projetos é muito fraca;
- A visibilidade dos problemas existentes é muito fraca;
- Clientes sempre pedem mais do que realmente precisam;
- Muita gente envolvida e pouca gente comprometida;
- Os projetos de software **são caros** e, ainda em sua maioria, **não alcançam o sucesso**.

Uso das Funcionalidades

Frequência de utilização das funcionalidades nos softwares



64% do software nunca ou raramente é utilizado

20% do software é realmente útil

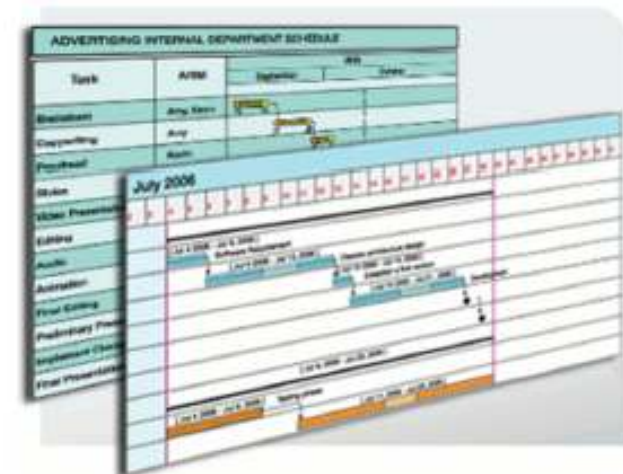


Processos empírico x Processos definidos

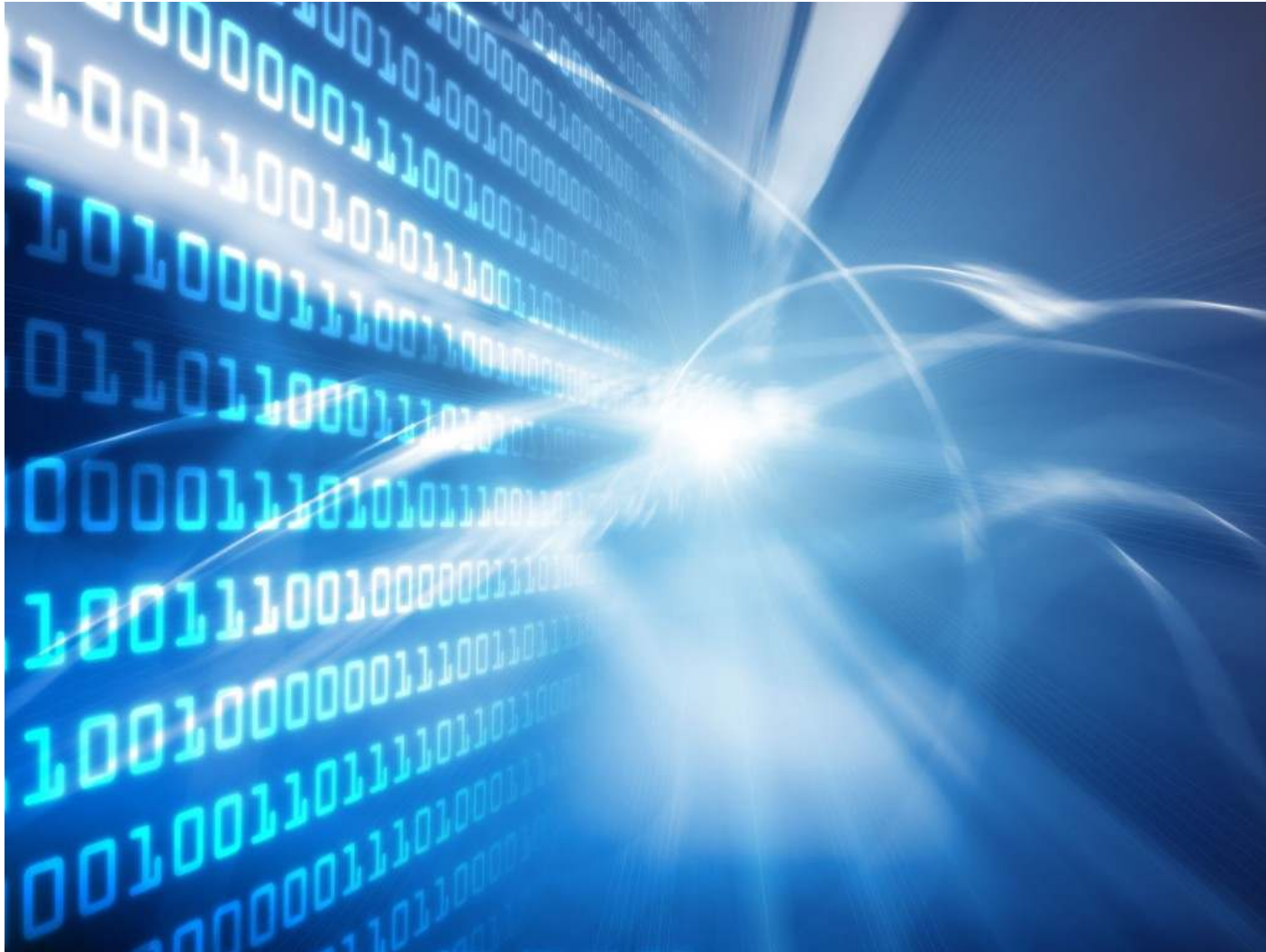


Processos Definidos??

- Modelo de processo definido com **mecanismos subjacentes claramente entendidos**;
- Sucessão de atividades claramente **definidas e lineares**;
- **Capacidade de estimar** tempos de execução de cada atividade;
- Normalmente gera as **mesmas saídas de produto**, quando ocorre **as mesmas entradas**.



Processos empírico x Processos definidos



Processos Empíricos?



- **Complexos, caóticos** ou seus detalhes ainda são **desconhecidos**;
- Atividades podem ser cíclicas e tem **durações** com **muitas variações**;
- É **difícil estimar tempos** de execução ou mesmo definir atividades a serem realizadas.

Desenvolvimento rápido de software



- Devido à rápida mudança dos ambientes de negócio, os negócios devem responder às novas oportunidades e à competição.
- Isso requer software e desenvolvimento rápido, e a entrega é, freqüentemente, o requisito mais crítico para sistemas de software.
- Os negócios podem estar dispostos a aceitar um software de baixa qualidade se a entrega rápida e a funcionalidade essencial for possível.

"Manifesto Para o Desenvolvimento Ágil de Software"



- reunião entre 17 gurus da comunidade de desenvolvimento
- Realizada entre os dias 11 e 13 de fevereiro de 2001
- Estação de esqui nas montanhas de Utah, Estados Unidos.

Manifesto Ágil (Princípios)

- **Indivíduos e interações** => mais importantes que *processos e ferramentas*.
- **Software funcionando** => mais importante do que documentação completa e detalhada.
- **Colaboração com o cliente** => mais importante do que negociação de contratos.
- **Adaptação a mudanças** => mais importante do que seguir o plano inicial.
- Evento ocorrido em 2001

WebSite: <http://www.agilemanifesto.org>

Metodologia Ágil

Envolvimento do cliente	Clientes devem ser profundamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema.
Entrega incremental	O software é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada incremento.
Pessoas, não processo	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar sem processos prescritivos.
Aceite as mudanças	Projete o sistema para acomodar mudanças.
Mantenha a simplicidade	Elimine a complexidade do sistema.

O que é agilidade?



- Equipe ágil
- Modificações
- Facilidade em alterar

Problemas com métodos ágeis



- ❑ Difícil manter o interesse dos clientes no processo
- ❑ As equipes podem ser inadequadas para o intenso envolvimento que caracteriza os métodos ágeis
- ❑ A priorização de mudanças pode ser difícil onde existem múltiplos stakeholders
- ❑ A manutenção da simplicidade requer trabalho extra
- ❑ Problemas nos contratos

Metodologias ágeis

(agile software development ecosystems - ASDEs)

- ❑ **XP (eXtreme Programming)**
- ❑ DSDM (Dynamic Systems Development Method)
- ❑ Família Crystal
- ❑ ASD (Adaptive Software Development)
- ❑ **SCRUM**
- ❑ FDD (Feature-driven development)
- ❑ LD (Lean Development)
- ❑ Open Source

Obs: Todos os seus autores com exceção do autor de LD e OpenSource participaram do Manifesto Ágil e portanto possuem princípios em comum.

XP (EXTREME PROGRAMMING)



XP (eXtreme Programming)

- Projeto C3 (Chrysler) - Kent Beck, Ward Cunningham and Ron Jeffries (1996)
 - ▣ <http://www.xprogramming.org>
- Valores:
 - ▣ Comunicação
 - ▣ Simplicidade
 - ▣ Feedback
 - ▣ Coragem
- 12 Práticas:
 - ▣ Pair Programing, Refactoring, Simple Design, Test-driven development
 - ▣ Collective Ownership, Coding Standard, Continuous Integration, Sustainable Pace
 - ▣ Customer tests, Whole Team, Planning Game, Small Releases, Metaphor

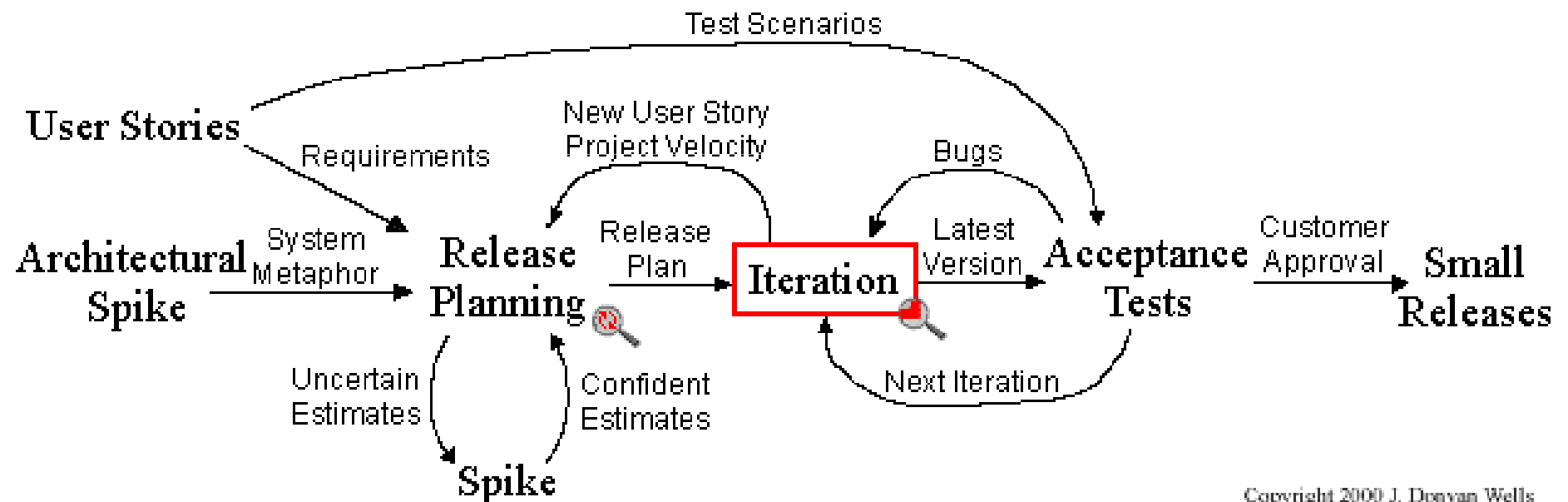
XP (eXtreme Programming)

□ 12 Práticas:

1. Processo de Planejamento (“Planning Game”)
2. Releases Curtos
3. Metáfora
4. Projeto(Design) Simples
5. Testes
6. Refactoring
7. Programação em Pares
8. Propriedade Coletiva do Código
9. Integração Contínua
10. Semana de 40 Horas
11. On-Site Customer (Cliente sempre presente)
12. Padrões de Codificação



Extreme Programming Project

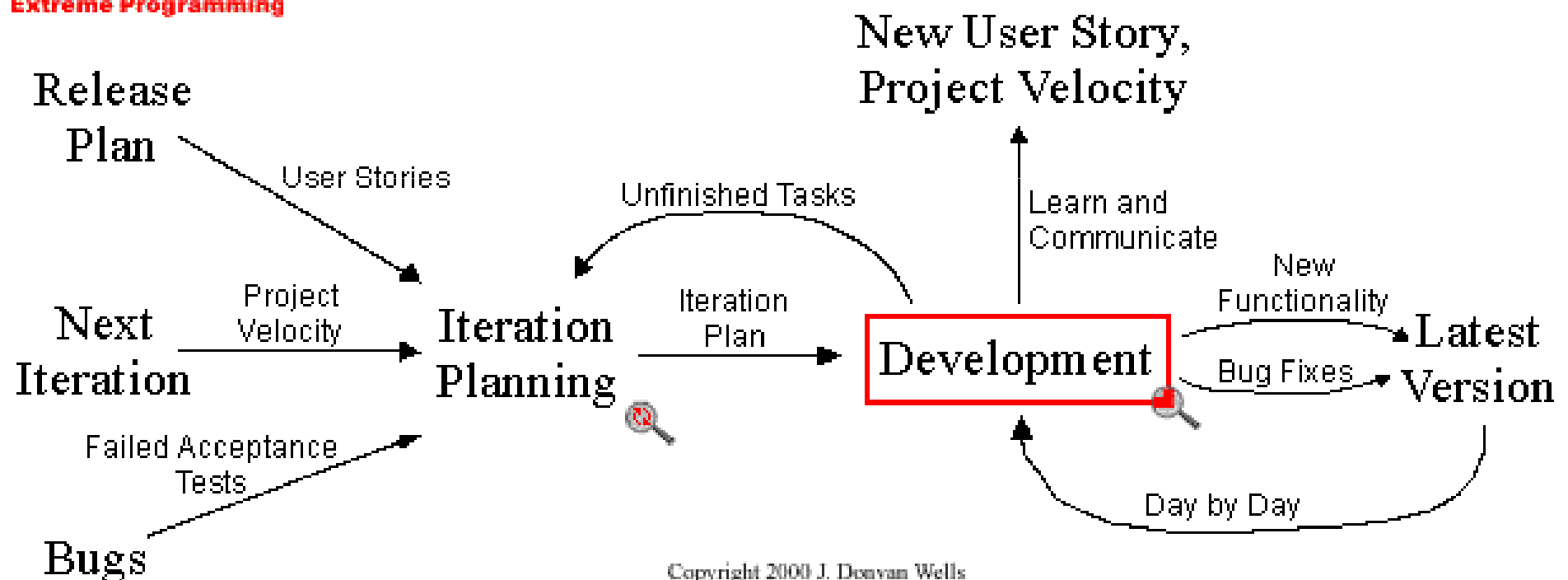


Copyright 2000 J. Donovan Wells

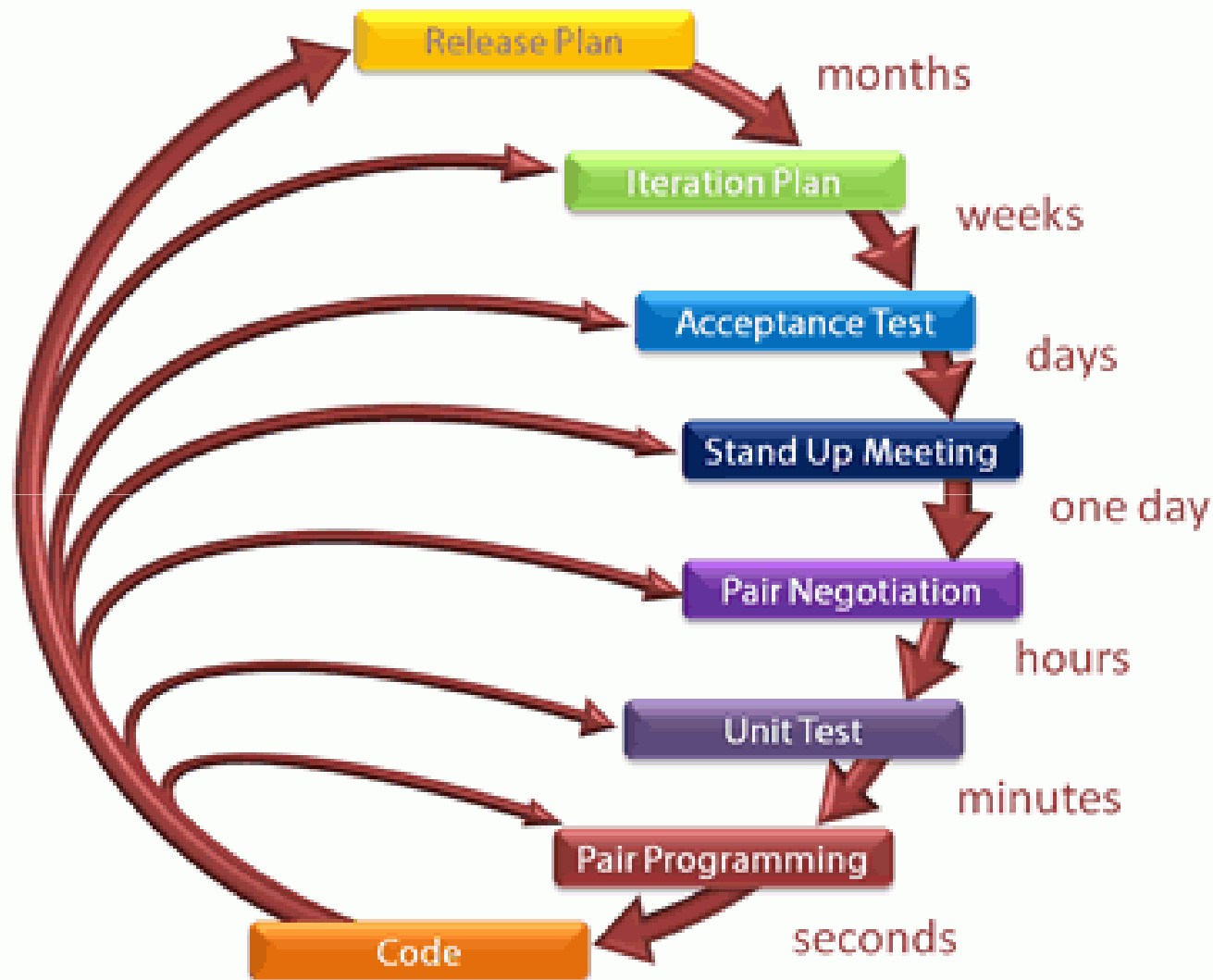


Iteration

Zoom Out



Extreme Programming Planning/Feedback Loops



© J. Donovan Wells

Papéis no XP



SCRUM

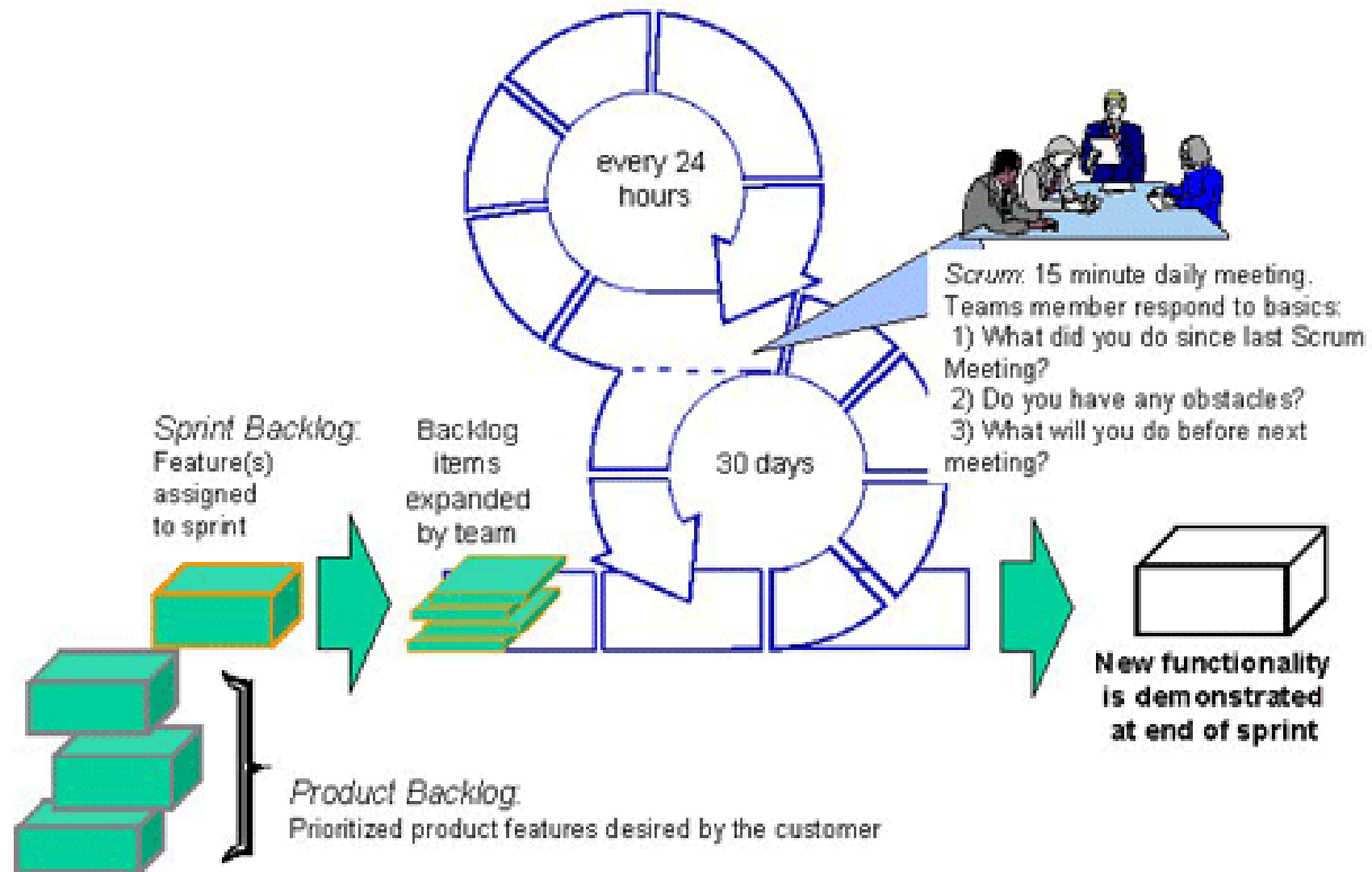


Rugby



SCRUM

- Jeff Sutherland, Ken Schwaber (1993)
 - <http://www.controlchaos.com/>
- Sprints de 30 dias
 - Estabilizar requisitos em cada iteração
- Scrum (reunião de status) diária (15 min)
 - Guia o desenvolvimento daquele dia
- Foco em gerência e tracking
 - Pode ser combinado com métodos mais prescritivos (ex: XP@scrum)



XP X SCRUM

- SCRUM com princípios semelhantes ao XP:
 - Equipes pequenas
 - Requisitos instáveis ou desconhecidos
 - Iterações curtas para prover visibilidade ao desenvolvimento.
- SCRUM com dimensões diferentes de XP:
 - Scrum divide o desenvolvimento em sprints de 30 dias e reuniões diárias de 15 minutos.
 - As equipes são formadas por pessoas com competências diferentes: projetistas, programadores, engenheiros e gerentes de qualidade.
 - Scrum possui um mecanismo de informação de status atualizado continuamente e a divisão de tarefas é explícita.
- SCRUM e XP são complementares pois Scrum prove práticas de gerenciamento enquanto que XP prove práticas integradas de engenharia de SW.

BACKLOG

Lílian

Product Backlog



- *É o coração do Scrum. É aqui que tudo começa.*
- *Uma lista de requisitos, histórias, Coisas que o cliente deseja, descritas utilizando a terminologia do cliente.*
- *Nós as chamamos de histórias, ou algumas vezes apenas de **itens do backlog**.*

As histórias incluem:

- **ID – Uma identificação única, apenas um número com autoincremento.**
 - Isso é para evitar que percamos o controle sobre as histórias quando nós mudamos seus nomes.

- **Nome – Um nome curto e descritivo para a história. Por exemplo,**
 - “Ver o histórico de transações”. Suficientemente claro para que os desenvolvedores e o *product owner* entendam mais ou menos sobre o que estamos falando, e claro o bastante para distingui-la das demais histórias. Normalmente de 2 a 10 palavras.

As histórias incluem:

- **Importância – a pontuação de importância dessa história para o *product owner*.**
 - *Por exemplo 10. Ou 150. Mais pontos = mais importante.*
- Deve-se evitar o termo “prioridade” já que prioridade 1 é tipicamente interpretado como “prioridade mais alta”, o que fica feio se mais tarde você decidir que algo é ainda mais importante. Qual pontuação de prioridade esse item deveria receber? Prioridade 0? Prioridade -1?

As estórias incluem:



- **Estimativa inicial –**

- As estimativas iniciais da equipe sobre quanto tempo é necessário para implementar aquela estória, se comparada a outras estórias. A unidade é pontos por estória e geralmente corresponde mais ou menos a “relação homem/dias” ideal.

As histórias incluem: Estimativa Inicial

- 1. Pergunte à equipe “se vocês puderem ter o número ideal de pessoas para esta história (nem muitas, nem poucas, normalmente duas), e se trancarem em uma sala cheia de comida e trabalharem sem distúrbio algum, após quantos dias vocês apresentarão uma implementação pronta, demonstrável e testada?”
 - Se a resposta for “com 3 pessoas trancados em uma sala levará aproximadamente 4 dias” então a estimativa inicial é de 12 pontos por história.

As histórias incluem: Estimativa Inicial



- 2. O importante não é ter estimativas absolutamente precisas
 - ▣ Por exemplo: dizer que uma história com 2 pontos deverá gastar 2 dias, mas sim obter estimativas relativas corretas (por exemplo, dizer que uma história com 2 pontos gastará cerca da metade de uma história com 4 pontos).

As histórias incluem: Estimativa Inicial

- **Como demonstrar – Uma descrição em alto nível de como a** história será demonstrada na apresentação do sprint. Isso é simplesmente uma simples especificação de teste. “Faça isso, então faça aquilo e então isso deverá acontecer.”
- Se você pratica TDD (desenvolvimento orientado a testes) essa descrição pode ser usada como pseudocódigo para o seu código de teste de aceitação.

As histórias incluem:



- **Notas – quaisquer outras informações, esclarecimentos, referências a outras fontes de informação, etc. Normalmente ágil bem breve.**

Exemplo: Product Backlog

PRODUCT BACKLOG (exemplo)					
ID	Nome	Imp	Est	Como demonstrar	Notas
1	Depósito	30	5	Logar-se, abrir a página de depósito, depositar R\$ 10,00, ir para a página do meu saldo e verificar que este aumentou em R\$ 10,00.	Precisa de uma diagrama UML de sequência. Não é necessário se preocupar com criptografia por enquanto.
2	Verificar seu próprio histórico de transações	10	8	Logar-se, clicar em "transações". Fazer um depósito. Voltar para transações, verificar se o novo depósito é listado.	Usar paginação para evitar consultas muito grandes ao banco de dados. Projetar de forma similar à página de visualização de usuários.

Campos adicionais - estória:

- **Track (Tilha/Rastro)** – *uma categorização básica dessa estória, por exemplo, “back office” ou “otimização”. Dessa forma, o product owner pode facilmente filtrar por todos os itens de “otimização” e definir sua prioridade para baixa, etc.*
- **Componentes** – **Geralmente são incluídos campos na forma de “checkboxes” em um documento no Excel, por exemplo “base de dados, servidor, cliente”.** Aqui a equipe ou o *product owner* podem identificar quais componentes técnicos estão envolvidos na implementação dessa história. Isto é útil quando se tem várias equipes de *Scrum*, como por exemplo uma equipe de *back office* e outra de cliente, assim facilitando a decisão de cada equipe na escolha das estórias.

Campos adicionais - estória:



- **Solicitante** – o *product owner* pode querer manter o **rastro de** qual cliente ou o *stakeholder* que solicitou o *item*, para poder fornecer um *feedback* sobre o *progresso* desse *item*.
- **ID do bug** – se houver um sistema separado para registro de erros (*bug tracking*), como nós fazemos com o *Jira*, é útil manter a rastreabilidade da estória com um ou mais erros reportados sobre ela.

Backlog – nível de negócio



- Colocar “Adicionar índice na tabela de eventos” no backlog?

Backlog – nível de negócio

- Ao invés de colocar “**Adicionar índice na tabela de eventos**” no backlog?
- A equipe é normalmente melhor adaptada para descobrir *como resolver algo*, assim o *product owner* deve *focar-se nos objetivos de negócio*.
- Então nós reformulamos a estória nos termos do objetivo subjacente (“**acelerar a pesquisa de eventos no formulário**”). A *descrição técnica original* acaba virando uma nota (“**Indexar a tabela de eventos poderia resolver isso**”).

PAPÉIS DO SCRUM

Lílian

Papéis do SCRUM

